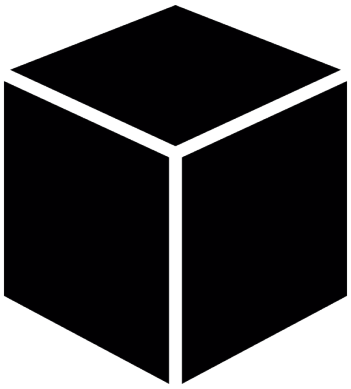
Vrije Universiteit Amsterdam
Accenture

How do you explain that?

An assessment of black box model explainers



Author
Marnix Maas

February 18, 2020

Abstract

Recent developments in machine learning have created methods to provide human interpretable explanations for any predictive model. This study compares model-agnostic explainers Lime and Shap in two experiments with real-world and synthetically generated data. Shap is advocated to provide explanations with highest quality according to theory on additive feature attribution methods. However, the experiments in this study suggest that Lime still competes with Shap, specifically on decision tree models. Secondly, a novel explainer evaluation metric is tested and deemed inadequate in its current implementation.

Contents

1 Introduction **1**

2 Literature review **3**

3 Methods **6**
3.1 Additive feature attribution methods 6
3.2 Perturbation 7
3.3 Lime 8
3.4 Shap 9
3.5 Parzen windows 12
3.6 Faithfulness metric 12
3.7 NDCG 13
3.8 Synthetic data generation 14

4 Experimental setup **15**
4.1 Real-world data 16
4.2 Lime experiment 5.2
"Are explanations faithful to the model?" 16
4.3 Experiment 5.2 | Improvements 17
4.4 Lime experiment 5.3
"Should I trust this prediction?" 19
4.5 Experiment 5.3 | Improvements 20
4.6 Experiments with synthetic data 20

5 Results **22**

6 Conclusion **31**

7 Discussion & Future work **32**

8 Appendix **36**
8.1 Lime 36
8.2 Shap 37
8.3 Supplementary results 38

1 Introduction

Machine learning models that produce unexplainable prediction have recently caught considerable attention. Not even the most practised data scientists can expose exactly why their deep learning model decided the way it did [7] [11]. Still, the number of predictive models implemented grows each day. Most of which are so difficult to interpret, we tend to refer to them as black boxes.
In light of this problem, advances have been made to methods that provide an explanation for any model. Two of these so-called post hoc model-agnostic explainers are Lime and Shap. However complicated the predictive algorithm, these explainers intend to provide insight into all individual decisions. The model explainers simplify the model's decision into a human understandable explanation.
Still, the literature on model explainers is divided: arguments are put forward that the use of such methods is flawed or not desirable, while others encourage their use or are improving the explainers. I suggest that this division has occurred because the (non-)quality of explainers is difficult to measure; little consensus exists in the evaluation of model explainers [12] [13]. As a consequence, the quest for advancing model explanation methods has been impeded. Making things worse, the more simplified the explanation, the more inclined it is to omit crucial information. If these challenges can be surmounted the improved accuracy of black box models can be utilized without sacrificing explainability.
Capabilities in model explanation methods are especially valuable for companies that market their machine learning expertise. Accenture advises other companies in opportunities with predictive algorithms and/or implements models to, e.g., improve efficiency. They can show how models provide value for other company, but will not be able to completely explain how predictions are formed when using black box models. Model explanation methods occupy this niche, as they allow Accenture to maximise value creation while maintaining the ability to explain any implemented model to the client.

This research aims to evaluate and compare recently proposed Lime and Shap. A baseline for these explainers is an older method named 'Parzen windows'. Since there is little consensus in the evaluation of model explainers [12] [13], the comparison is realised using two simulated user experiments previously proposed by Ribeiro et al. [1]. These experiments are constructed so that the working of the predictive algorithms is known. Hence, the expla-

nations formed for each prediction can be evaluated. In addition, I propose several modifications to the implementation of both experiments to improve evaluation of explainers. The experiments are performed on three real-world datasets and four sets of synthetically generated data.
A second aim in this research is to analyse and assess a novel explainer evaluation metric called 'faithfulness' proposed by Arya et al. [5]. If it can be shown that this evaluation metric is reliable, it could provide the desired basis for comparing model explainers.

This study firstly introduces recent literature on the subject of post hoc model-agnostic explanation methods. Then, all implemented methods and their usage in the experiments are expanded upon. Figures and tables form an overview of the experiments' results and will thereafter be discussed. Lastly, the comparison of Lime and Shap is concluded and a discussion is established on findings in this research.

2 Literature review

Recent developments and research on post hoc model-agnostic explainers are firstly discussed. Thereafter, a novel method to assess the quality of such explainers is covered. Lastly, tendencies against the use of explanation methods are considered.
It has been suggested that the terms explainability and interpretability should not be used as synonyms, as they may be different in nuance[1]. However, in this study these terms are treated as equivalent.

Post hoc model-agnostic explainers

In recent years, many advances in linear post hoc model-agnostic explainers have been made. The quest for such methods seems to come from the trade-off between model performance and explainability. Most recently best performing machine learning algorithms, while having better accuracy over previous methods, are hard or impossible to interpret. For example: neural networks and ensemble methods. As a result, a new field in model explainability has arisen. Most versatile are those methods that can be applied to any model - model-agnostic - and after the model has been trained - post hoc -.
To this end, Ribeiro et al. [1] have introduced Lime in 2016. Lime being a Local Interpretable Model-agnostic Explainer. In short, the method creates a sparse linear explanation of an instance from the data that is locally faithful. The explanation consists of a maximum of K features, a user specified number, which are assigned an attribution to the prediction. The explanation is additive: all feature attributions (and a base value) sum up to approximately the model's prediction for that instance. The explanations are locally faithful in the sense that the explanation should resemble the model in the vicinity of the instance, but is not guaranteed to be globally faithful. In addition, the paper argues that Lime fulfills above mentioned properties using some simulated experiments.
An interesting addition to the explanation model is the notion of coverage. Their SP-Lime algorithm will find a set of the most representative explanations. That set of explanations should optimally describe the global behaviour of the algorithm.
In a work by Lundberg and Lee [2] all additive feature attribution methods,

[1] Interpretability would suggest a degree to which the model's output can be predicted. While explainability covers the degree of being able to explain to a human the inner mechanics of an algorithm.

like Lime, are joined under a single definition. They unite Lime, DeepLIFT, layer-wise relevance propagation and Shapley value approximators (like in [11]). Then, it is shown that only Shapley values [10] [14] can satisfy three desirable properties that enforce a unique solution for an additive feature attribution explanation. According to Shapley's theorem, that unique solution is optimal. Shortly, these properties cover accuracy and consistentcy of the explanations. Lastly, Lundberg and Lee propose a new method named Shap Kernel to approximate Shapley values with improved sample efficiency over previous methods. It should be noted that Shap has brought the theoretical basis of Shapley values from game theory to additive feature attribution methods. However, it is assumed that features are independent and the explanation model is linear.
To counter these assumptions, an improved method was proposed by Aas et al. [3] in 2019. They suggest that the assumption of independency of features can produce faulty explanations. Thus, they propose an improvement of the Shap Kernel method to handle dependent features. Since the assumption of independency is only necessary in one step of calculating Shap values, they suggest relaxing that assumption. They found that for non-linear models the improved method outperforms Shap. A drawback of allowing dependencies is that explanations become harder to interpret. Multiple dependent features can only be properly interpreted as a group, rather than individually. Additionally, this extension to Shap increases computation time.
In addition to extending Shap for dependent features, an extensive overview of the Shap Kernel approximation method is given since the original paper [2] does not fully describe the implementation. An accurate description is given of how the sample efficiency is improved over previous Shapley value approximations.

Aside from the creation of and improvements to model explainers, new metrics have been proposed to evaluate them. Previously, the quality of model explainers has mostly been evaluated based on the inclusion of features: recall. How many of the features used by the model can the explainer find? While it can offer insight, recall alone crucially omits the attributions provided by explainers. A novel 'faithfulness' metric has been suggested by Arya et al. [5]. The metric aims to evaluate the quality of an explanation and its attributions with the use of correlation. They have suggested a second evaluation metric named 'monotonicity', although that relies on similar evaluation of correlation. As of this date, no research has been done to evaluate their explainer faithfulness metric.

Disadvantages to post hoc explainers

The advances in explanation methods are compelling and show potential for decreasing the trade-off between model interpretability and performance. However, some works have put forward that using post hoc explainers may not be beneficial.

In a recent paper from November 2019 by Slack et al. [7] a framework was created to fool Lime and Shap into producing faulty explanations. In short, the framework creates a clearly biased model and hides that bias by abusing the perturbations on which explainers rely. The explanations produced by the Lime and Shap seem unable to reproduce that bias. It should be noted that explainers are susceptible to "adversarial attacks" as Slack et al. suggest. Thus, if one has wrong intentions, Lime and Shap can be fooled into faulty explanations.

In some domains, for specific models, post hoc explainers have been shown to miss the most relevant feature of an instance. One such example is from Camburu et al. [4] for explaining a neural network on a natural language processing task. Their goal is to create (a framework for) an evaluation test for post hoc explanatory methods from the perspective of feature-selection. Their evaluation only considers the inclusion of features, unlike the faithfulness measure from Arya et al. While Camburu et al. conclude that post hoc explainers can miss the most relevant feature of a prediction, their framework was only tested on a single model for a single task. As they note in their conclusion, their framework could be applied to other tasks and areas.

3 Methods

This section will describe techniques used in the simulated user experiments. Firstly, an intuition of 'linear' model explanation methods is given. Model explainers Lime and Shap are discussed in detail, as they will be evaluated in the experiments. The Parzen windows explainer from [9] is used as baseline for Lime and Shap and will be shortly reviewed. In order to improve evaluation of explainers, two evaluation metrics are defined: faithfulness and NDCG. Lastly, the approach to synthetic data generation is considered. All experiments also are applied to the synthetic data for which the level of noise and redundancy can be controlled.

3.1 Additive feature attribution methods

The explanation methods used in this paper fall into the category of 'linear explainers' or 'additive feature attribution explanation methods'. Linear comes from the fact that these explainers rely on sparse linear models for their explanations. The following sections will expand on the exact methods for Lime and Shap. In short, all explanations start at the average probability that the prediction belongs to that class, namely: 'base rate'. The explainer assigns an attribution to all features it includes, up to a provided maximum K features. Adding the attributions/impact of all features to the base rate results in (approximately) the algorithm's prediction. A formal definition of an additive feature attribution explanation is given by:

$$g(z') = \phi_0 + \sum_{i=1}^{M} \phi_i z'_i,$$

where g is the explanation model that includes only features selected in z'. With $z' \in \{0,1\}^M$ is the selection of features (maximum of K) for that explanation. The features are indicated by $i \in \{1, ..., M\}$ where ϕ_i defines their individual attribution and base rate is denoted as ϕ_0.
An example of additive feature attributions might be useful before going into detail. Consider Figure 1 from Lundberg's repository[2]. Firstly, the figure shows what the explainers are allowed to see. A model-agnostic explainer only has access to the data and the model's output, but not to the black box model itself. In the example, the base rate starts at 0.1. The positive attributions displayed in green for the features Age, BP and BMI increase the probability by a total of 0.6. Whereas the negative attribution displayed

[2]https://github.com/slundberg/shap. Figure has been modified.

in blue of feature Sex decreases the probability by 0.3. The sum of the base rate and all attributions comes out to be 0.4, which is (approximately) the algorithm's prediction.

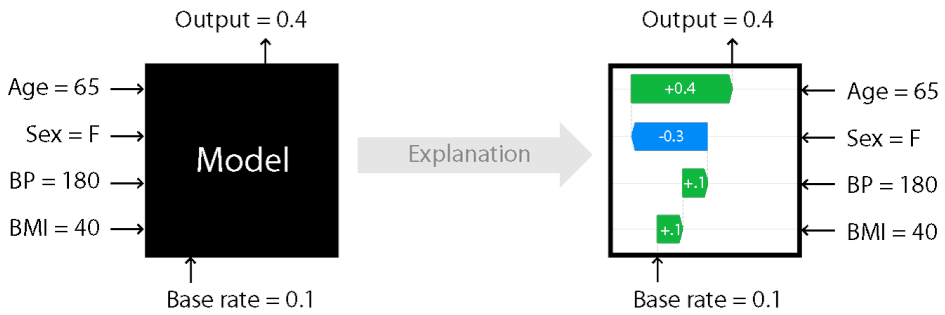


Figure 1: An example of additive feature attribution for a single prediction. The base rate (mean probability of that class) of 0.1 summed with the total of all feature attributions, an additional 0.3, results in the output value of 0.4.

3.2 Perturbation

Post hoc model-agnostic explainers in this study rely on feature perturbation in order to provide explanations while treating the classifier as a black box. To measure the influence of a single feature on the prediction, it is taken out of the instance, i.e., the initial value. But, most machine learning methods do not allow absence of value for a feature. Thus, the value is replaced in order to mask its impact. The value can be replaced by the most common/average value from the background dataset or a value of 0. A sensible choice for that replacement depends on the data. The process of masking a feature from an instance is called perturbation. An advantage is that perturbation works with any classification model.
Perturbation or the masking of features can also be done for multiple features of an instance. For Lime and Shap a perturbed sample of an instance x is denoted as z or z'.

3.3 Lime

The Local interpretable model-agnostic explanations method, or Lime for short, has been proposed by Ribeiro et al [1]. In short, Lime aims to provide a human interpretable explanation that is locally accurate to, and can be applied to any, machine learning model.
Firstly, interpretability, model-agnostic explainers and (local) accuracy are characterized:

- An explanation is interpretable if it can be easily understood by humans. For Lime specifically, it assumes that a sparse linear model is interpretable. Sparsity meaning that the linear explanation incorporates a small number of features that would allow for an easily readable explanation. E.g., a linear explanation with only 5 features should be quickly understandable for a human.

- The method is model-agnostic. The explanation method does not make any assumptions about the underlying predictive model. Therefore, it can be applied to any machine learning model.

- The explanations are locally accurate to the predictive model. According to Ribeiro et al. the explanation "must correspond to how the model behaves in the vicinity of the instance begin predicted". Therefore, the explanation should resemble the model in a local manner, but that does not imply global accuracy.

Formally, Lime has that g is an explanation with a limited number of features included. The complexity of g for linear explanations is measured by the number of included features and is denoted as $\Omega(g)$. G is the family of interpretable models (for example, a small decision tree would also apply). Let $f(x)$ be the probability that x belongs to a certain class according to model *f*. The proximity (vicinity) between instance x and z is $\pi_x(z)$, where z is a perturbed sample of instance x. Then Lime is defined by:

$$\xi(x) = \underset{g \in G}{argmin}\ L(f, g, \pi_x) + \Omega(g).$$

Lime finds the explanation $\xi(x)$ that is the most locally accurate, while still being interpretable. A formal definition of L is included in Appendix Section 8.1. The user can specify the complexity budget or maximum allowed number of features K. When the explanation becomes larger than K, the complexity constraint $\Omega(g)$ will be infinite. This enforces the explanation to be smaller than K. The explanation will look as described in Section 3.1.

How the method works intuitively is shown in Figure 2. The explanation (displayed in dashed lines) is constructed with the weighted perturbed samples around the instance. The perturbed samples are depicted as red crosses and blue circles. The background colour indicates the decision boundary of the model. This figure aims to show that an explanation can be locally, but not globally, accurate to the decision boundary of the model.

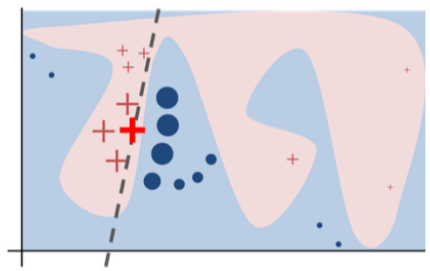


Figure 2: Source [1]. Intuition of Lime method: the linear explanation in dashed lines is constructed with weighted perturbed samples around the instance.

3.4 Shap

A unified approach to model explanation methods named Shap, has been proposed by Lundberg and Lee [2]. In their work they aim to show that if a linear explanation is created for a model, Shap values are the most consistent and computationally viable. The name Shap originates from Shapley additive explanations.
Shapley values are a theorem from game theory, but they are used in the Shap explainer. Firstly, Shapley values are described. Thereafter, the use of Shapley values in the Shap explainer is considered.

Shapley values

Shapley values are a method for distributing the total payout of several games over n persons, depending on their contribution to the payout for each game [10]. According to the theorem, Shapley values are the optimal assignment of payout since they adhere to a set of desirable properties. The following section will cover these properties, as they are distinctly for additive feature attribution methods. A formal definition of the properties is included in Appendix Section 8.2. Young has shown that a solution that

adheres to those desirable properties is unique [14]. Thus, according to these theorems, Shapley values are the unique set of additive attributions for an optimal linear explanation.
Molnar concisely describes Shapley values: "A prediction can be explained by assuming that each feature value of the instance is a 'player' in a game where the prediction is the payout. Shapley values – a method from coalitional game theory – tells us how to fairly distribute the 'payout' among the features".[3]

Unified definition - additive feature attribution methods

Shap Kernel is based on the Lime method, but with different choices for the weighting kernel π (distance metric) and regularization term Ω. For Lime, these parameters are chosen heuristically, whereas the Shap Kernel method defines these parameters according to the Shapley theorem. Formally, in Lundberg and Lee's theorem 2, the Shapley kernel is defined as:

$$\Omega(g) = 0,$$

$$\pi_{x'}(z') = \frac{M-1}{(M\ choose\ |z'|)|z'|(M-|z'|))},$$

where M is the number of features, the number of non-zero features in z' is $|z'|$ and x' is the set of all perturbations of the data. With this definition of the weight kernel and regularization, the estimated values should adhere to 3 desirable properties. Whereas the heuristic choice for Lime may result in a violation of local accuracy and consistency.

A formal definition of the desirable properties is given in Appendix Section 8.2. Intuitively, the properties can be explained as:

- Local Accuracy: The explanation exactly matches the predicted probability by the model for an individual. The mean explanation summed with all explained feature attributions is the same as the output of the model.
- Missingness: Features not included in the explanation do not have any attribution to the explanation.
- Consistency: Feature attribution should not decrease if the features' input is kept the same or increased, while the other inputs are unchanged.

[3] https://christophm.github.io/interpretable-ml-book/, chapter 5.9

A unique additive feature explanation model follows from these properties:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!}\,[f_x(z') - f_x(z' \setminus i)].$$

With i indicating a specific feature.

Shap Kernel

The complexity of calculating Shapley values depends on the number of features. The exact calculation of Shapley values becomes computationally intractable for a large number of features. Since the exact calculation is computationally expensive, Lundberg and Lee have proposed Kernel Shap to approximate them. Under the assumption that features are independent and the model is linear, Shap values can be calculated with higher sample efficiency than previous Shapley equations. The improved sample efficiency is described in detail in a different paper in Section 2.3 by Aas et al. [3]. They elaborate that the calculation of Shap values is made more efficient by performing part of the matrix calculations once for several explanations at a time, instead of once for each explanation. In addition to the model-agnostic Shap Kernel method, Lundberg and Lee have introduced several model specific methods for calculating Shap values even more efficiently. These include Deep Shap for deep learning models and Tree Shap for tree based models. However, for this research only true model-agnostic methods are considered.

In summary, Shap introduces a faster method to approximate Shapley values. According to the theory, only Shapley values adhere to a triplet of desirable properties for an explainer. Like the Lime method, Shap will produce an explanation with feature attribution for up to K features.

3.5 Parzen windows

The Parzen windows technique is an approach to estimate the probability density function of a specific point without knowing the underlying distribution. A region around the point is used to estimate the value of probability density, which is where the name Parzen windows originates. The method will also be referred to as 'Parzen' in this study. Baehrens et al. [9] describe how Parzen windows can be used to explain individual classification decisions.
They define the Bayes classifier:

$$g^*(x) = arg \min_{c \in \{1,\dots,C\}} P(Y \neq c | X = x)$$

where C is the number of classes in the classification problem and $P(X, Y)$ is some unknown joint distribution. Then, the explanation vector of a data point x_0 is the derivative to x at $x = x_0$. Formally noted as:

$$\zeta(x_0) := \frac{\partial}{\partial x} \; P(Y \neq g^*(x) | X = x)\Big|_{x=x_0}.$$

With $\zeta(x_0)$ a M-dimensional vector, with the same length as x_0. The explanation is formed by the largest (absolute) feature attributions in the vector $\zeta(x_0)$, up to K features. Note that this is similar to the explanation vector $\xi(x)$ definition from Lime.

3.6 Faithfulness metric

The quality of an explainer depends on the interpretability offered to the user as well as the faithfulness to the model. These notions may have an opposing effect. A simpler explanation may not fully resemble the model, as it cannot capture its full extent.
Current post hoc explanation methods use a sparse linear explanation. They offer the same level of interpretability as long as they have the same number of features. Thus, they could easily be compared based on some notion of local accuracy to the model.
Currently, there is no standard metric to measure that local accuracy. However, Arya et al. [5] have introduced an inconveniently[4] named fidelity metric called: 'faithfulness'. This metric measures the quality of explainers,

[4]Literature on this subject tend to use local accuracy, fidelity and faithfulness as synonyms. This study carefully uses those terms to prevent confusion with the faithfulness metric.

rather than human evaluation being the golden standard. This study aims to determine if the metric is viable for assessing the quality of explainers.

The faithfulness metric expresses the quality of an explanation as correlation between model predictions and feature attributions: in order of feature importance the feature of an instance is perturbed (replaced by the background value), then both the model's predicted probability and the feature attribution are recorded. That process is repeated for all features for which a feature attribution exists. The faithfulness metric ϕ is then defined as the negative Pearson correlation ρ between the vector of feature attributions Θ and the vector with the model's prediction probabilities $\mathbf{p}$:

$$\phi = -\rho(\Theta, \mathbf{p}).$$

The higher ϕ the better the quality of an explainer (beware not to confuse this 'ϕ' with feature attributions from previous definitions). Intuitively, the model's prediction probability should decrease when a feature with positive attribution is removed. Thus, the faithfulness metric aims to show to what degree that intuition is followed. In other words, the method scores explainers for attribution values that have the same impact as the model when perturbing a feature.
The method has a drawback, as this metric uses correlation, it is not defined for small explanations. It is not possible to calculate correlation when the length of Θ and $\mathbf{p}$ is 1, since correlation is not defined for a point. When the length of Θ and $\mathbf{p}$ is 2 the correlation is always either 1 or -1. The metric would always assign either the best or worst possible score to the explanation. This is not desired behaviour for such a metric. Nonetheless, explanations would often consist of more than two features. For these explanations the faithfulness metric may still provide insight by scoring the intuition as described above.

3.7 NDCG

Normalized Discounted Cumulative Gain or NDCG is a measure of ranking quality. The metric is mostly applied in information retrieval. Search algorithms are scored by their ability to retrieve the most relevant documents in order. I propose to use this metric in the evaluation of model explainers. This popular technique assesses the explainers by the features it retrieves along with the ranking of features. Formally, NDCG is defined as:

$$NDCG_p = \frac{DCG_p}{IDCG_p},$$

with

$$DCG_p = \sum_{i=1}^{p} \frac{rel_i}{log_2(i+1)},$$

and

$$IDCG_p = \sum_{i=1}^{|REL|} \frac{2^{rel_i} - 1}{log_2(i+1)},$$

where rel_i indicates presence of individual feature i and $|REL|$ is the list of features ordered by importance. Intuitively, the metric will assign a score between 0 and 1, comparing the explanation's ranking of features to the model's true rank of features. A value of 1 indicates a perfect ordering.

3.8 Synthetic data generation

For the two experiments synthetic data is generated. With synthetic data the number of features, dependencies, noise and other factors can be controlled. Then, the robustness of model explainers can be evaluated given the alterations to the data.
Data is generated using the make_classification function from sklearn[5]. The package provides a method to generate a dataset with user specified modifications. Firstly, the user defines the number of informative features. Then, a number of redundant features can be specified. These redundant features are a random linear combinations of the informative features from the dataset. In addition, noise is created by replacing the target variable with a randomly selected target output. The following section will elaborate on other parameters used in the data generation process.

[5] https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_classification.html

4 Experimental setup

There is hardly consensus for the evaluation of model explainers [12]. Hence, the comparison is done via simulated user experiments. These experiments are constructed so that the working of the predictive algorithm is known and thus, can be used to quantitatively assess the explainers. The experiments are based on those presented in a paper by Ribeiro et al. [1]. Their code is provided in an online repository[6]. This is the basis for the experiments in the current paper. Accordingly they will be named Lime experiment 5.2 and 5.3 for the original and Experiment 5.2 and 5.3 for the modified versions in this study. Minor adjustments have been made to run this experiment in Python 3.7 rather than 2.7.

In order to improve the measurement of explainer quality, several adjustments for the implementation of both experiments are proposed. Those improvements include the two explainer quality metrics: faithfulness and NDCG. Additionally, both revised experiments are applied to synthetically generated data. Since that data can be manipulated to test the explainers in their handling of redundancy and noise. The following subsections will describe all experiments in detail. Code for the current study is available at: https://github.com/marnixm/lime_experiments.
While Aas et al. [3] suggest that using Shap with an extension for dependent features may be beneficial, its explanations are not as simple as Lime and Shap provide. Its explanations can only be properly interpreted as clusters of dependent features. For the current study only true additive feature attribution methods are considered. Thus, Shap with the extension for dependent features is not included in the experiments.
The original Lime experiments include a random and greedy explainer. However, they only provide inclusion of features, but not feature attribution. Since feature attribution is crucial to explainers, the greedy and random explainer are insufficient as baseline. Hence, they too have been excluded from the experiments.
As mentioned in the faithfulness Section 3.6, the metric is not defined for explanations of two or fewer features. Instances to which this applies have been omitted from the experiment's results.
All explainers are provided a maximum budget of $K = 10$ features for their explanations. The Parzen explainer may find explanations of larger size, unlike Lime and Shap. Therefore, the Parzen explanations will consist of

[6] https://github.com/marcotcr/lime-experiments

only the K most important features (as was the choice in the original Lime experiments).

4.1 Real-world data

The data consists of product reviews from Amazon.com. The data has been used for several studies, initially by Blitzer et al. [6]. A review is labelled with a binary positive or negative outcome. The features in this dataset are the words used in the reviews for each domain. Accordingly, the predictive models are performing sentiment analysis. For the current study, the datasets on books, DVDs and kitchen products have been used.
All three datasets contain approximately 20,000 features and 2000 rows of data. The data is split into a train and test set of respectively 1600 and 400 rows.

4.2 Lime experiment 5.2 "Are explanations faithful to the model?"

In this experiment, machine learning methods are used that are interpretable by themselves. Namely sparse logistic regression and decision trees. However, these models are only allowed to use a maximum of $K = 10$ features for each row in the test set. Thus, for all instances, a golden set of features is known. In this experiment, the writers aimed to show that their explainer can find the features used by the model.
A comparison is made of the golden set of features for each instance to the explanations. Thus, scores for recall of golden features from the model can be calculated. Precision would also be an interesting metric to consider. However, the models are asked to provide an explanation of 10 features. If the model would use fewer than 10 features itself, the explainer could never reach a precision of 1. Therefore, only recall was considered.
The sparse logistic regression model uses L1 penalty, where the penalty parameter is increased until a maximum of 10 features for each row is used. Similarly, the decision tree model is only allowed to use a maximum of 10 features for the path through the tree for an instance.
It should be noted that while Section 5.1 from the Lime paper [1] describes the use of L2 regularization for the linear regression in their experiments, it is actually L1 regularization that was implemented in their repository. Both Shap and Lime are provided with a budget of 15,000 samples.
Lastly, beware that the title of this experiment is not to be confused with the faithfulness metric. In [1] the terms local accuracy, faithfulness and fidelity

are used as synonyms. The title of this experiment actually advertises that the explainer should use the same features as the models themselves: recall.

4.3 Experiment 5.2 | Improvements

Number of golden features per instance

The sparse linear regression model from Experiment 5.2 is provided with an increasingly high penalty until a maximum of 10 features are used for all instances. More precisely, the features used by model (Θ) and the features provided by the explainer (ξ) are for **all** instances: $|\Theta \cap \xi| < 10$. It is this intersection that is referred to as golden set of features. In practise, the sparse linear regression model is provided with such a high penalty, that this intersection has an average of 2-3 features. Thus, the explainer is allowed to provide 10 features, whereas on average 2-3 features are used per instance. As a result, high recall numbers of over 90% are reported.

Instead, it is suggested to find models with an average (rather than maximum) of 10 used features for each instance. For the calculation of recall and faithfulness, only the 10 most important features are considered. In the original experiment, the explainers were allowed a larger budget to retrieve all features ($|\xi| > |\Theta|$). Now, the explainers will have to retrieve a number of features that is more frequently equal to their budget ($|\xi| \approx |\Theta| \approx K$). By increasing the number of golden features per instance, the test should be more challenging for the explainers.

However, for the decision tree model it is not reasonably possible to increase the number of golden features. Initially, the model uses 200+ distinct features in the complete tree. However, an instance walks a specific path among the branches of the tree in order to reach a prediction. This path will in all likelihood not reach every feature, thus the golden set of features is smaller than the 200+ from the complete tree. For the real-world datasets used, the average number of golden features is actually only 1.4 to 1.5. It is possible to increase the size of the tree. However, the tree would be specifically trained so that it uses an increased number of features. In contrast to a normal machine learning setting where the tree would be optimized on, for example, each split. In other words, the larger decision tree would not represent a model build in any real world situation. Therefore, it was decided not to modify the decision tree model. Only for the logistic regression model will the number of golden features be increased.

Table 1 shows an overview of the average number of golden features for the improved experiment.

	Books	DVDs	Kitchen
Logistic regression	9.5	9.8	7.1
Decision tree	1.5	1.4	1.4

Table 1: Experiment 5.2 improved | Average number of golden features per instance of the real-world data

Ranking of explained features

So far only the inclusion of features has been considered. Even though all explanation methods in this study also provide feature attribution: a degree of impact on the prediction. Whereas the faithfulness metric attempts to evaluate each attribution, I would argue that the ordering of variables should already provide a better measurement than recall alone. An explainer is even better if it can find the golden set of features and rank them in order of importance. Hence, I propose to take ranking of variables into account. To this end, the NDCG is used to calculate the quality of explanations.

For the logistic regression model features are ordered by their (absolute) coefficient. The order of features for the decision tree model is determined by their global variable importance.[7] In addition, to calculate the NDCG both the explainer and instance must be of the same length. The NDCG is calculated only for the n most important golden features per instances, where n is the minimum number of features in the explainer or instance. More precisely: $n = min(|\Theta|, |\xi|)$ for each instance. However, if the explanation provides fewer features than the model it would not be penalized. To counter this shortcoming the NDCG score is multiplied with the recall for that instance. If the explanation is too short, it will be penalized by the recall score. If the model uses fewer features than the explanation, only the number of features in the model is considered, due to the cut-off at length n. After all, the explainer should not have returned additional features for its explanation.

To summarise, it is proposed to consider the NDCG score for evaluation of explanations, since the ranking of variables is taken into account. To adjust for explanations of unequal length, we penalize using recall.

[7] The global importance is still combined with the golden features for that instance. The Discussion section will expand on this decision

4.4 Lime experiment 5.3 "Should I trust this prediction?"

Trustworthiness of the explainers is assessed by considering the impact of features on the model and the explanation. In this experiment it is assumed that the user can identify a number of features that are not trustworthy and that the user would not want to include in the model. For each instance in the dataset 25% of features are sampled to be untrustworthy. It is then compared how the predictions of the model and explainer change by removing the attribution of the untrustworthy features. In other words, the user 'discounts' the attribution of untrustworthy features.

Discounting for the model is done by replacing untrustworthy feature values with the background value. The modified instance is passed through the classifier for a new predicted probability. In case of the explainer the initial prediction is formed by the base rate summed with all explained feature attributions. Discounting for the explainer is the initial prediction subtracted with attributions of each feature that is untrustworthy in that instance.

For both the model and explainer, the experiment will compare the initial prediction with the prediction after discounting untrustworthy features. If the untrustworthy features change the prediction of the model, the explainer too should switch its prediction. It is pointed out that the classification problem is binary. Thus, a change of prediction is defined by going from a negative to a positive classification, or vice versa.

Considering the change of prediction results in a vector of trusted and mistrusted instances; precision and recall can be calculated from these vectors. In the original paper, the F1 score was reported for two datasets.

Trustworthiness can be tested for any classification model. Five models are considered in this experiment: sparse logistic regression (LR), nearest neighbours (NN), random forest (RF), support vector machine (SVM) and decision tree (Tree). Model parameters are equal to [1]. Only the solver for the logistic regression was changed to 'lbfgs' since the previous solver is no longer supported.

For this experiment the faithfulness metric is not implemented. This experiment chooses an arbitrary number of untrustworthy features and discounts them from the explanation. By design, the explanation before and after discounting should not differ in quality. Hence, the performance of explainers in this experiment cannot be measured with the faithfulness metric.

4.5 Experiment 5.3 | Improvements

Penalization of local inaccuracy
In the original paper it was decided to compare whether both the model and explainer changed predictions due to the removal of untrustworthy features. However, the model and explainer should in the first place have the same prediction. While this is always the case for Shap values due to local accuracy guarantees, Lime and Parzen explanations may initially be wrong.[8] That had not been taken into account in the original experiments. To adjust for this flaw, a new 'local accuracy' score is introduced. For convenience, the score will be referred to as accuracy. The accuracy of the explainer is decreased if the prediction, before removal of untrustworthy features, is not equal to the model. The improved experiment proposes a penalization of the F1 score if the local accuracy property is violated. Experiment 5.3 will present this adjusted F1 score.

4.6 Experiments with synthetic data

Synthetic data is generated so that modifications to the data can be specified. The controlled modifications may expose weaknesses of the explainers. The improved experiments 5.2 and 5.3 are repeated using the synthetic data. Several adjustments were made to allow the use of synthetic data.
Firstly, to provide Lime explanations the Lime tabular package is implemented. The original experiments actually do not use the package, but an implementation specific to text classification. Thus, the package is implemented for use with the synthetic data. The standard Lime kernel has been used. In addition, both Lime and Shap now require a background dataset for perturbations. Lime is provided with the complete background dataset, while the background dataset for Shap is summarised using K-means clustering to keep computations feasible (as Lundberg suggests in his repository[9]). For this experiment the data is summarised in 10 clusters. The Parzen windows explainer uses two parameters, these are set equal those of the Books dataset.

All datasets are constructed to include 10 informative features, the same number of features the explainers are allowed to provide (budget $K = 10$). From these 10 features, either 0 or 15 redundant features are created. Extra

[8] Note that the model has a binary outcome, predictions side with positive or negative at the threshold of 0.5.

[9] https://slundberg.github.io/shap/notebooks/Iris%20classification%20with%20scikit-learn.html

random (useless) features are added so that the total number of features is 50. Datasets of 50 features keep these experiments computationally viable, though for future work a larger number of features may prove a more challenging test. The noise parameter fluctuates between 0.05 or 0.3. To conclude, four datasets are generated with low to high amount of noise and redundancy. All datasets will have 2000 rows and are similarly split into a train and test set of respectively 1600 and 400 features.
As with the improved Experiment 5.2 the average number of golden features for the generated data is reported in Table 2.

	Redun: 0 Noise: 0.05	Redun: 15 Noise: 0.05	Redun: 0 Noise: 0.30	Redun: 15 Noise: 0.30
Logistic regression	10.0	10.0	10.0	9.0
Decision tree	11.9	12.9	14.2	11.7

Table 2: Experiment 5.2 improved | Average number of golden features for each instance of the synthetic data. Redundancy is notated as 'Redun'.

5 Results

This section will present the results of all simulated user experiments described in the previous section. These experiments aim to measure the performance of explanation methods.
Firstly, the original Lime experiment 5.2 is presented. Secondly, the results from the improved experiment are shown. Thirdly, we repeat the improved experiment with synthetic data. Then, the faithfulness metric is tested in adherence with previous results. The outcome of (Lime) Experiments 5.3 will thereafter be presented.

Lime experiment 5.2

In this experiment small interpretable models are trained so that a golden standard of features is known. It should be noted that the average number of true features, is only between 2-3. Whereas the explainers are given a budget of 10 features for their explanations. Explanations are generated for each instance in the test dataset. The average recall of features is shown in Figure 3. The y-axis indicates the machine learning model used. The x-axis displays the dataset. Even though the explanation methods are model-agnostic, the figure shows varying results.

Figure 3: Experiment 5.2 | Real-world data evaluated on recall

Considering the logistic regression model, it can be seen that Shap is able to retrieve all features for these datasets. Lime produces the second highest recall for two of the three datasets: Books and DVDs. Only on the Kitchen dataset is Lime outperformed by Parzen.[10] In contrast, for the decision tree model Lime has dominant recall over the other explainers. Shap retrieves the second highest number of features, still more than the baseline for all datasets.

Experiment 5.2 | Improved

For this experiment, the number of golden features used for each instance has been increased. So that the average (rather than maximum) number of golden features is nearer to 10, an overview is shown in Table 1. For some instances the logistic regression uses over 50 features in the golden standard. For these instances only the 10 most important features are considered. As a result of increasing the number of golden features the presented recall scores are lower in comparison to the original experiment. Note that the number of golden features for the decision tree has not changed.
The recall scores for this improved experiment are shown in Figure 4.

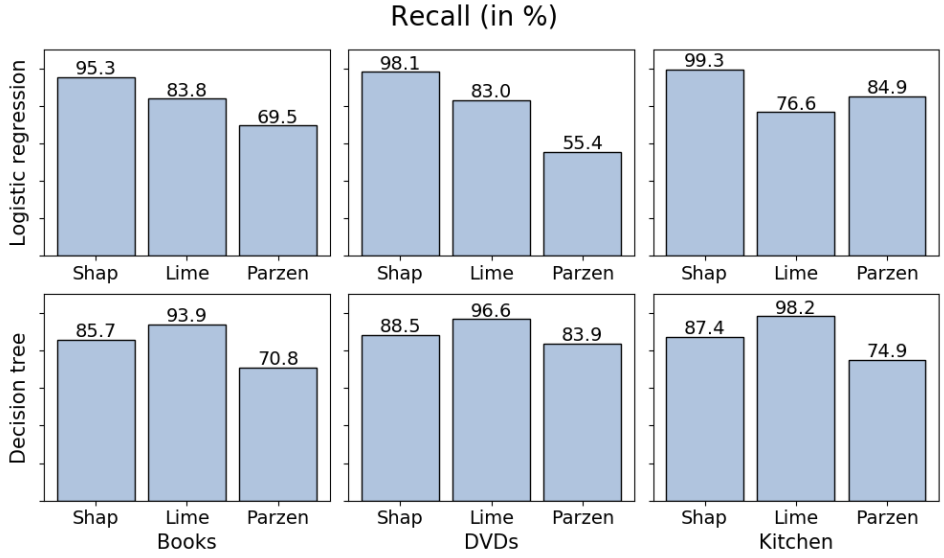


Figure 4: Experiment 5.2 improved | Real-world data evaluated on recall

[10]Interestingly, that particular dataset was excluded from the results section of [1].

The figure shows that for the logistic regression model all recall scores are lower compared to the original experiment. This would be expected since the explainers now have to retrieve a number of features (approximately) equal to their budget. The relative performance of the explainers has not changed. Though, it should be noted that Shap no longer has a perfect recall. Still, Shap dominates the results for the logistic regression model and Lime for the decision tree model. In addition, Parzen's score on the logistic regression model for the Books dataset decreased by more than 20%.
The NDCG scores for the explainers are presented in Figure 5. Not only is feature inclusion measured, but the ranking of features is now taken into account as well.

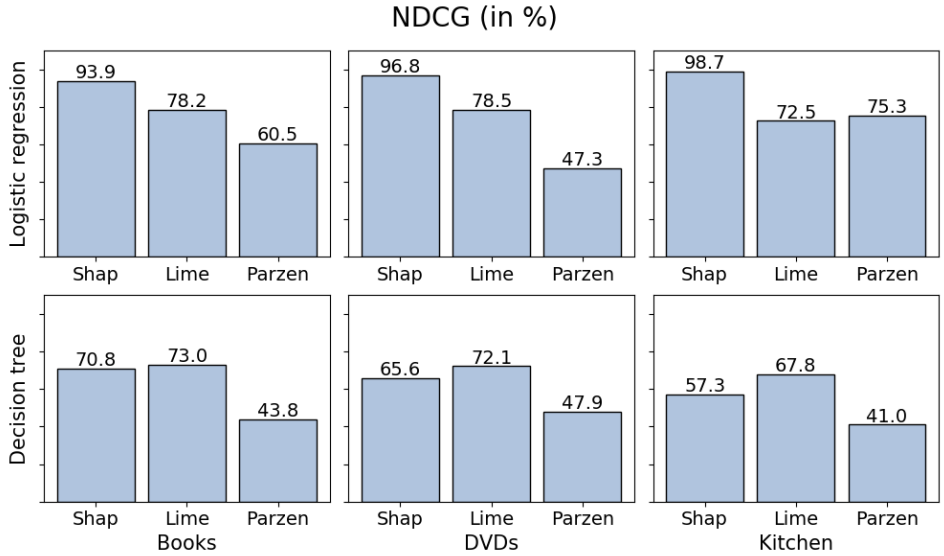


Figure 5: Experiment 5.2 improved | Real-world data evaluated on NDCG

Firstly, consider the logistic regression model. All scores displayed are naturally lower since the ranking of features is measured. Lime and Parzen's scores have now decreased below 80%, where Shap still manages NDCG scores of 90% and above. The decision tree models show greater decrease in explanation quality. The highest NDCG score reported is 73%. The figure shows that Lime still outperforms Shap on decision trees, whereas they both perform better than the baseline.

Experiment 5.2 | Synthetic data

Synthetic data is generated so that the underlying dependencies and noise can be controlled. This data allows for a better assessment of the explainers as the impurities of a real-world dataset are not present.
Recall of golden features is shown in Figure 6. On the logistic regression model it shows comparable results for Lime and Shap, all scores are above 85%. Though, Shap maintains the highest results. Considering the decision tree model it is interesting to mention that Shap produces a slightly higher score than Lime on the synthetic dataset with high noise and redundancy. Lime still reaches highest recall for the other synthetic datasets. Where Parzen's score has previously shown lowest results, for this experiment the recall even decreases below 15%. This happens for the data with redundant features.

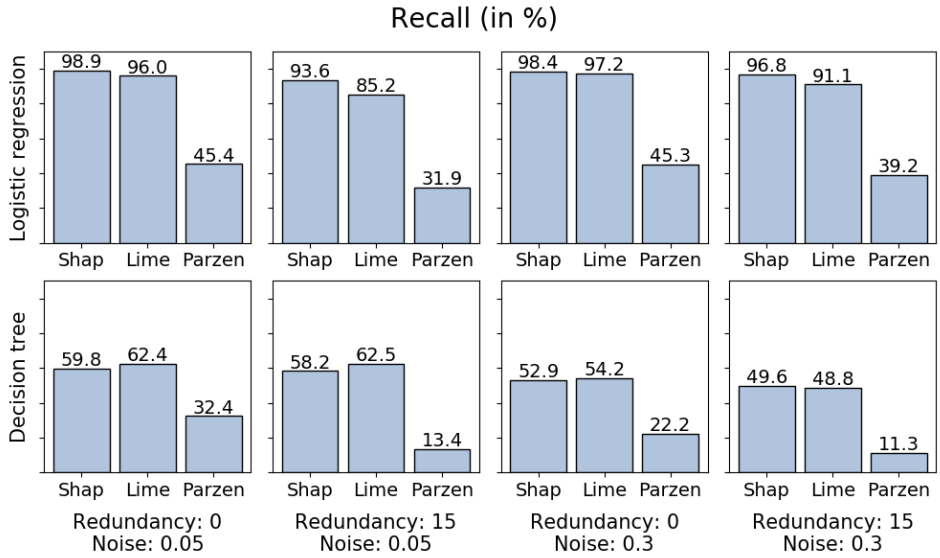


Figure 6: Experiment 5.2 improved | Synthetic data evaluated on recall

Figure 7 presents the NDCG scores for generated data. For the logistic regression model and data with low redundancy Lime actually performs best, where previously Shap had dominated. Though, once redundant features are introduced Shap performs marginally better than Lime. For the decision tree Lime still produces the best explanations, followed by Shap. Parzen produces the lowest results. Again, its lowest scores are presented when redundant features are included.

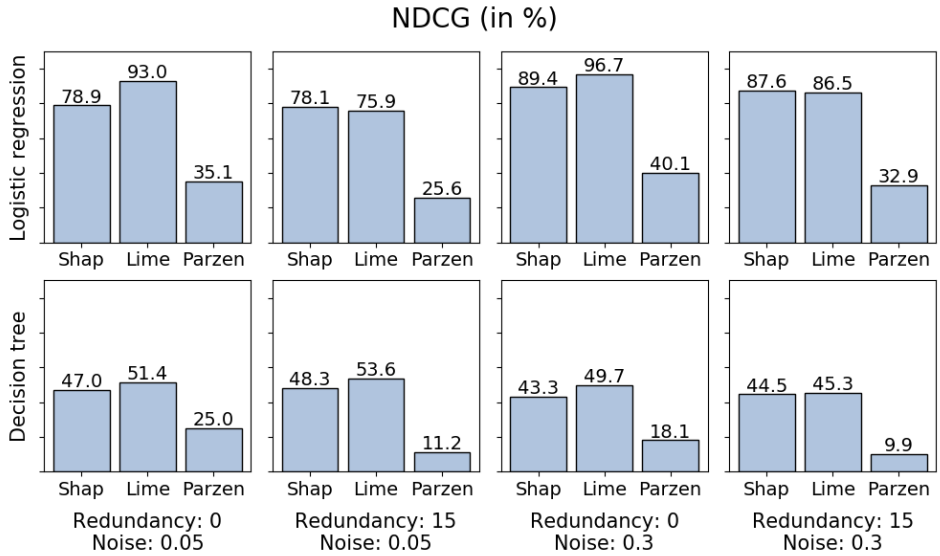


Figure 7: Experiment 5.2 improved | Synthetic data evaluated on NDCG

I suggest that the NDCG scores present a better image of explainer quality, since it allows the measurement of ranking of features. Furthermore, when NDCG is corrected with recall, the measurement is also able to deal with explanations that involve fewer features than the provided budget K. Lastly, the experiments measured on recall alone show high performance on both models. NDCG identifies that the correct features are indeed obtained, but their attributions are not properly assigned. Hence, all reported NDCG scores present a lower but more accurate score.
In summary, for this experiment Shap presents the best results for the logistic regression model. Lime performs best with the decision tree model. According to those results, an argument is put forward that Shap's assumption of independent features is punished (as has been suggested in [3]) by models that allow that dependency: decision trees in this case. This is supported by a reduced recall score for Shap with the decision tree model.

Experiment 5.2 | Faithfulness metric

The average faithfulness scores for this experiment (original, improved, synthetic data) are respectively shown in Figures 8, 9, 10. Note that in theory, the higher the faithfulness score, the better the explainer is.
The figures demonstrate that Shap has dominantly higher faithfulness scores

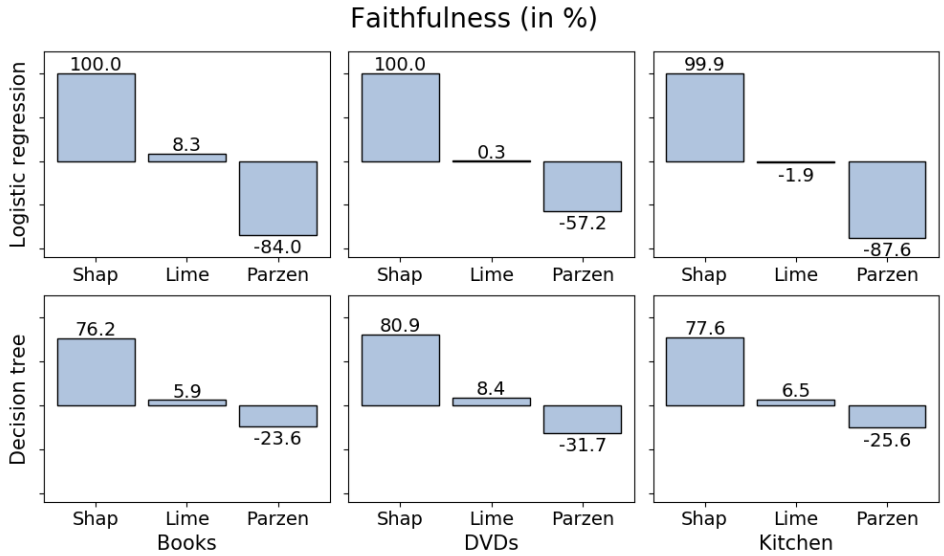


Figure 8: Experiment 5.2 | Real-world data evaluated on faithfulness

than the other explainers on all datasets, all above 75%. In second place comes Lime with scores fluctuating from approximately -14% to 12%. Lastly, the baseline Parzen often produces lowest scores. Most often achieving the lowest faithfulness score by a large margin.

However, upon further investigation into the specific scores of the faithfulness metric for each explainer, the metric produces extreme outcomes. For a large number of instances the faithfulness scores are nearly 1 or -1. These extreme outcomes seem to originate from the budget provided to the explainers. Since the explainers are asked to find an explanation of size $K = 10$, they will assign features, not present in the instance, a tiny attribution. Especially when that instance includes fewer than 10 golden features. The model's prediction does not change, because those features are not included. The explainer however, has assigned a tiny attribution. The Pearson correlation, on which faithfulness relies, does not distinguish these points as negligible. As a consequence, these explanations are assigned scores of approximately 1 and -1. Especially for Lime the faithfulness scores fluctuate between these extremes. As a result, the faithfulness scores average out in the middle: close to 0. While having in mind that the explainers are requested to provide a K-sized explanation, it is argued that this behaviour of the faithfulness metric is not desirable.

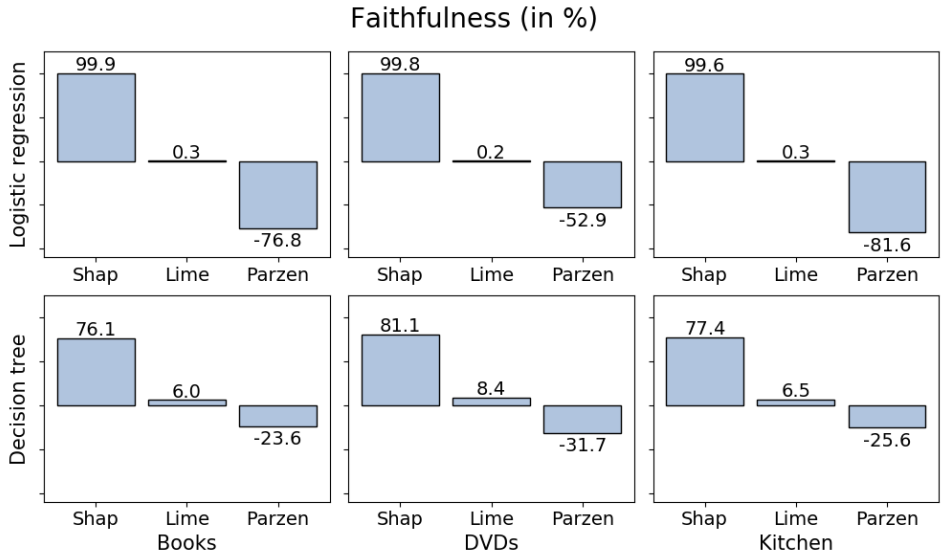


Figure 9: Experiment 5.2 improved | Real-world data evaluated on faithfulness

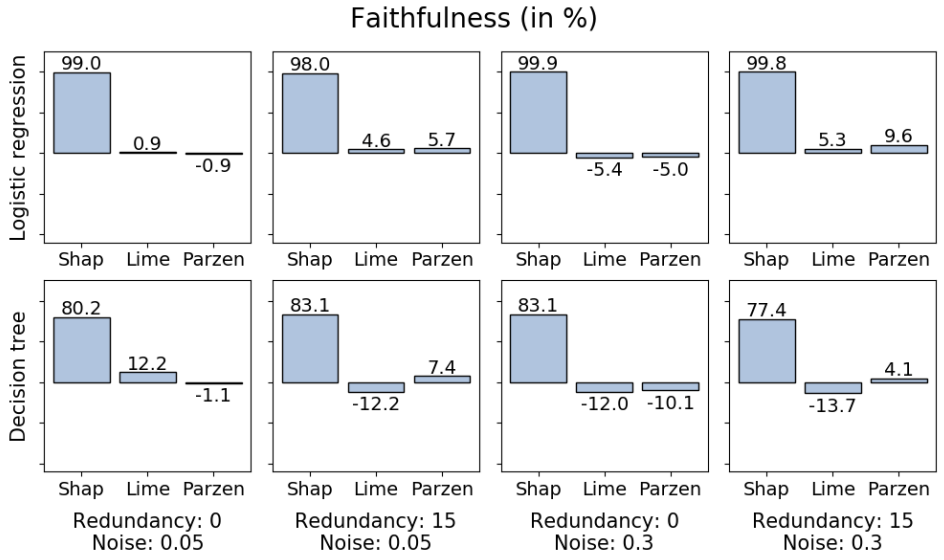


Figure 10: Experiment 5.2 improved | Synthetic data evaluated on faithfulness

Experiment 5.3 | Improved

For each instance of the data an arbitrarily selected 25% of features is deemed untrustworthy. The prediction of the model and explainer are compared before and after untrustworthy features are discounted. The adjusted F1 score is reported in Table 3.

Shap is able to produce the highest adjusted F1 score for all three datasets on logistic regression, nearest neighbours, support vector machine and random forest models. However, Lime performs marginally better on the decision tree model. Both Lime and Shap clearly outperform the baseline Parzen. The baseline scores in the range of 35% - 66%. Interestingly, similar results have shown in experiment 5.2 where Lime performs better than Shap on decision tree models.

Individual scores for recall, precision and accuracy measure are included in Appendix Section 8.3. Shap produces a perfect score on accuracy. That is due to the local accuracy property, which Shapley values are guaranteed to satisfy. The accuracy scores do give a penalty to the Lime and Parzen method, as their initial classification is different from that of the predictive model. Parzen is most heavily punished by its accuracy scores. The results show that Parzen often wrongly classifies the initial prediction. As a result, for this experiment Parzen does not compete with either Lime or Shap.

Adjusted F1 score (in %)

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	**96.8**	**92.3**	**98.5**	**96.7**	93.9	**96.5**	**87.0**	**98.5**	**95.9**	94.8
Lime	95.8	84.0	93.9	94.9	**97.3**	95.5	80.3	97.4	95.1	**97.7**
Parzen	53.7	61.1	59.6	65.6	54.3	53.5	47.5	47.8	55.5	48.7

	Kitchen				
	LR	NN	RF	SVM	Tree
Shap	**97.8**	**91.4**	**99.2**	**97.6**	95.0
Lime	97.4	82.8	98.3	97.5	**97.6**
Parzen	34.8	68.3	64.5	48.0	58.5

Table 3: Experiment 5.3 | Real-world data evaluated on adjusted F1 score

Experiment 5.3 | Synthetic data

Experiment 5.3 is repeated with synthetically generated data. A random selection of untrustworthy features is discounted from the explanations. The resulting adjusted F1 scores are presented in Table 4. It is reminded that the generated datasets are constructed with low to high amount of redundancy and noise.

Table 4 shows that Shap delivers the highest adjust F1 score for all datasets and models. In particular, the scores for the decision tree model are relatively higher than the Lime and Parzen explanations. Lime often produces second highest results, although exceptions exist where Parzen reaches second highest scores.

Appendix Table 8.3 displays the individual precision, recall and accuracy scores. Lime and Parzen are able to compete with Shap in terms of precision. However, Lime and Parzen are not able to reach similar scores for recall and accuracy. Hence, Shap has dominant performance when considering the combined adjusted F1 scores.

Adjusted F1 score (in %)

	Redundancy: 0 Noise: 0.05					Redundancy:15 Noise: 0.05				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	**98.2**	**94.8**	**98.6**	**97.4**	**93.6**	**98.3**	**96.3**	**98.3**	**98.2**	**93.1**
Lime	94.5	84.8	87.8	92.4	72.9	95.6	85.9	91.4	90.5	76.8
Parzen	81.5	79.8	80.4	75.4	71.3	85.2	86.9	89.2	88.7	72.4
	Redundancy: 0 Noise: 0.30					Redundancy:15 Noise: 0.30				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	**97.9**	**88.3**	**98.0**	**97.6**	**90.5**	**98.1**	**92.9**	**98.3**	**96.6**	**90.2**
Lime	91.5	70.1	87.0	90.5	62.8	89.0	78.5	90.7	84.3	60.6
Parzen	73.3	64.5	75.9	71.9	58.6	76.7	79.9	86.7	86.1	54.6

Table 4: Experiment 5.3 | Synthetic data evaluated on adjusted F1 score

6 Conclusion

The explainability of machine learning algorithms is receiving abundant attention in current literature. As of yet, the literature struggles to create unity in assessing and evaluating model explainers. This study proposes the comparison of additive feature attribution explainers via simulated user experiments with real-world and synthetic data.

Consider Figure 3. The original Experiment 5.2 (using real-world data) boasted a perfect score for Shap on the logistic regression model. While Lime's performance is highest on the decision tree model. Improvements to that experiment have been proposed, so that the budget for the explainers is better aligned with the number of golden features from the models. Additionally, the ranking of features is taken into consideration using the NDCG metric. Results from the improved experiment are displayed in Figure 4 and 5. All recall scores are lower when compared to the original experiment. The relative performance of the explainers is unchanged. Still, the NDCG scores suggest that Lime and Parzen are penalized harder than Shap once the order of features is included. Still, Lime marginally outperforms Shap on the decision tree model. Lastly, the experiment is carried out with generated data, see Figure 6 and 7. Recall scores of Lime and Shap are comparable for both models. Shap performs better on the logistic regression model, but highest scores vary between Lime and Shap for the decision tree. However, the NDCG scores firstly introduce Lime as the better explainer on the logistic regression model on data with low redundancy. When redundant features are introduced Shap marginally outperforms Lime. For the decision tree Lime slightly outperforms Shap.
Simulated user Experiment 5.3 has been modified using a (local) accuracy score. The resulting adjusted F1 scores on the real-world data are reported in Table 3. For all models other than the decision tree Shap delivers the best results. Again, for the decision tree Lime has better performance. Individual accuracy scores in Appendix Section 8.3 reveal that Lime and Parzen are penalized for initially wrong classifications.
Experiment 5.3 is repeated with synthetic data. Adjusted F1 scores are displayed in Table 4. Shap performs better than the other explainers on all datasets and models.

In summary, improvements to previously proposed user experiments aim to better evaluate the quality of model explainers. In contrast to the theory Lime frequently produces better explanations for decision tree models in

these experiments. Nevertheless, Shap still provides better explanations for the logistic regression model in Experiment 5.2 and all models other than the decision tree in Experiment 5.3 with real-world data. Experiment 5.3 with synthetic data was dominated by Shap's results. To conclude, according to the theory Shap is expected to produce explanations of best quality among the class of additive feature attribution methods. These experiments suggest that Lime, with its heuristically chosen kernel, may still prove a competitor to Shap in terms of explanation quality.
The faithfulness metric is tested as evaluation metric for linear model explainers. Its scores are presented in Figures 8, 9 and 10. The metric is unfit for evaluation of explanations with fewer than two features. Additionally, it shows undesirable behaviour even when the metric is defined for an instance. Thus, the faithfulness metric is deemed inadmissible for the evaluation of model explainers.
In conclusion, current explanation methods prove a powerful tool for providing interpretable insights into black box models. Those covered in this study can already be implemented by Accenture for machine learning projects. Though, it is noted that (relative) quality of explainers is yet inconclusive and it is advised to implement explanation methods with discretion.

7 Discussion & Future work

This section covers several assumptions and choices made in this study. In addition, suggestions for future improvements to these experiments and future work on model explainers are put forward.
The main goal in this work is to make advances to the evaluation of model explainers. As the literature has not yet reached consensus of how to measure the explainers' quality [12]. To this end, this research considers feature inclusion in the form of precision and recall. The proposed NDCG measure is able to include the ranking of features by their importance. However, NDCG can only be calculated if the true order of features is known. That is the case for logistic regression, the prediction for an instance is formed by the global coefficients. However, for the decision tree model only **global** feature importance is known. For experiment 5.2 the true ordering of features was considered using the global variable importance. To calculate the NDCG, that ordering is compared to the ordering of features in the explanation. While using the global feature importance is not perfectly representative, experiment 5.2 aims to show that including feature ranking is an improvement over just the recall scores.

The faithfulness metric was tested on three versions of experiment 5.2. This study has found that in its current implementation, the metric is not admissible for the evaluation of model explainers. Since the metric uses correlation, it does not show desirable behaviour for vectors with a length of two or fewer features. Additionally, the metric shows deviating behaviour when explanations (or changes in model predictions) are minuscule. Nonetheless, it is the only measure that actually considers each attribution value of model explanations. If these flaws can be overcome, the metric may become a robust model explainer evaluation metric. Furthermore, should any evaluation metric for model explainers be formulated (especially if it can integrate attribution), it would prove a substantial basis for the field of post hoc explainers.
Both experiment 5.2 and 5.3 have been applied to generated data, so that the degree of dependency and noise can be controlled. Data generation was implemented using sklearn's make_classification function. However, the generated data could have been controlled even further by using an option such as 'SymPy'. Data generation using symbolic expressions is described in this post by T. Sarkar[11]. With SymPy the explainers could be tested on their handling of noise and dependencies with increased control.
An important factor that has been omitted in this research is the speed and computational effort required for the explanation methods. While I would suggest that speed is not the most important element of model explainers, it could prove valuable considering that Lime's performance is comparable to Shap's in the experiments in this work. Note, that this is the case for Shap Kernel. Lime tabular was up to several times faster than Shap Kernel in the current implementation for an equal number of samples (with Shap's background data summarized in 10 clusters).

Considerations in the current study may already have exposed remaining challenges for model explainer In addition to those, I propose several improvements to and future work for model explainers.
Lime and Shap require the user to specify the number of features that should be included in the explanation. Given this length, the user should be able to interpret the explanation. Now imagine an instance where one additional feature could significantly decrease the inaccuracy of the explanation. Then, certainly, the user would want to include that additional feature, even if the

[11] https://towardsdatascience.com/random-regression-and-classification-problem-generation-with-symbolic-expression-a4e190e37b8d

maximum number of features is exceeded. Creating a test to determine the optimal number of included features or improvement of the explanation with each additional feature, would be a promising addition to model explainers.

Lastly, while this research is aimed at the use of model explainers, others suggest that explanations can be harmful due to their unexpressed inaccuracy. As mentioned in the literature review, linear explanations for highly non-linear models may not be sufficient. For some instances, it may simply not be (logically) possible to form an additive feature explanation. A proper explanation method should indicate that the inaccuracy to the instance is excessive, or even establish that an additive explanation is not possible. I propose that such an extension, while being challenging to design, would greatly increase trust in model explainers. A different approach is suggested future work in [1]: the local accuracy "can also be used for selecting an appropriate family of explanations from a set of multiple interpretable classes". In other words, the interpretable model that forms the explanation could be specifically chosen for the instance to increase accuracy. Two alternative explanation models include:

1. Shap extension from [3] that handles dependent features, though the concept may be applied to other additive feature explainers. A drawback is that explanations become harder to interpret, as they can only be considered as a cluster of dependent features.

2. I suggest that different interpretable models may be a more suitable alternative to the Shap extension in situations where explanations ought to be simpler. For example, a (small) decision tree instead of a sparse linear model for the explanations may provide a solution to non-linearity and simplicity.

Instead, as an alternative to utilizing a different explanation model, Robeiro et al. had suggested in [1] that the information of local (in)accuracy to an instance could be shown to the user. Then, the user might decide whether to endorse the explanation.
In the end, one desires from an explanation that it is locally accurate to the model and easy to interpret. However, increasing the length or complexity for better accuracy compromises in interpretability. Likewise, oversimplifying an explanation is prone to result in a less accurate explanation. Hence, in the field of model explainers a tension between interpretability and accuracy has formed. Promising future work would progress in both these concepts.

References

[1] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "Why should i trust you?" Explaining the predictions of any classifier.

[2] Lundberg, S. M., & Lee, S. I. (2017). A unified approach to interpreting model predictions.

[3] Aas, K., Jullum, M., & Løland, A. (2019). Explaining individual predictions when features are dependent: More accurate approximations to Shapley values.

[4] Camburu, O. M., Giunchiglia, E., Foerster, J., Lukasiewicz, T., & Blunsom, P. (2019). Can I Trust the Explainer? Verifying Post-hoc Explanatory Methods.

[5] Arya, V., Bellamy, R. K. E., Chen, P.-Y., Dhurandhar, A., Hind, M., Hoffman, S. C., Houde, S., Liao, Q., Luss, R., Mojsilovic, A., Mourad, S., Pedemonte, P., Raghavendra, R., Richards, J., Sattigeri, P., Shanmugam, K., Singh, M., Varschney, K., Wei, D. & Zhang, Y. (2019). One Explanation Does Not Fit All: A Toolkit and Taxonomy of AI Explainability Techniques.

[6] Blitzer, J., Dredze, M., & Pereira, F. (2007). Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification.

[7] Slack, D., Hilgard, S., Jia, E., Singh, S., & Lakkaraju, H. (2019). How can we fool LIME and SHAP? Adversarial Attacks on Post hoc Explanation Methods.

[8] Lundberg, S. M., Erion, G. G., & Lee, S.-I. (2018). Consistent Individualized Feature Attribution for Tree Ensembles.

[9] Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., Hansen, K., & Müller, K. R. (2010). How to explain individual classification decisions.

[10] Shapley, L. S. (1953). A value for n-person games.

[11] Štrumbelj, E., & Kononenko, I. (2014). Explaining prediction models and individual predictions with feature contributions.

[12] Doshi-Velez, F. & Kim, B. (2017). Towards A Rigorous Science of Interpretable Machine Learning.

[13] Gilpin, L. H., Bau, D., Yuan, B. Z., Bajwa, A., Specter, M., & Kagal, L. (2019). Explaining explanations: An overview of interpretability of machine learning.

[14] Young, H. P. (1985). Monotonic solutions of cooperative games.

8 Appendix

8.1 Lime

This section is supplementary to Methods Section 3.3. Remind that Lime finds the explanation that is the most locally accurate, while still being interpretable. The local accuracy to the explained instance is maximised for an explanation with a maximum complexity $K = 10$. For a linear explanation the local accuracy $L(f, g, \pi_x)$ is formally defined by:

$$L(f, g, \pi_x) = \sum_{z,z' \subseteq Z} \pi_x(z) \ (f(z) - g(z'))^2$$

where z and z' are perturbed samples.
The (heuristically chosen) kernel is defined as:

$$\pi_x(z) = exp\left(\frac{-D(x,z)^2}{\sigma^2}\right)$$

with D is some distance function with width σ. The distance function for text classification is cosine distance. Accordingly, the current study also implements cosine distance.

Perturbations
Additionally, the perturbations from the Lime tabular implementation in this study is expanded upon. Numerical data is perturbed by reverse scaling: a sample is taken from a normal N(0,1) distribution and scaled to the mean and deviation from the training data. Categorical features are sampled from their distribution in the training data.

8.2 Shap

This section is supplementary to Methods Section 3.4. In Lundberg's and Lee's paper [2] they show that only Shapley values adhere to a set of desirable properties for explanation methods. While an intuitive definition of these desirable properties is provided in Methods, the definitions are formally given by:
Property 1: Local accuracy

$$f(x) = g(x') = \phi_0 + \sum_{i=1}^{M} \phi_i x'_i$$

Property 2: Missingness

$$x'_i = 0 \Longrightarrow \phi_i = 0$$

Property 3: Consistency
If $f'_x(z') - f'_x(z' \backslash i) \geq f_x(z') - f_x(z' \backslash i)$ for all inputs $z' \in \{0,1\}^M$, then

$$\phi_i(f', x) \geq \phi_i(f, x)$$

Note that ϕ_i is the attribution of feature i. A unique additive feature explanation model follows from these properties:

$$\phi_i(f, x) = \sum_{z' \subseteq x'} \frac{|z'|!\,(M - |z'| - 1)!}{M!} [f_x(z') - f_x(z' \setminus i)]$$

8.3 Supplementary results

Experiment 5.3: Real-world data

The individual scores for precision, recall and accuracy for Experiment 5.3 on the real-world data are presented in the tables below.

Precision (in %)

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	95.2	88.0	98.6	95.5	92.3	94.5	81.7	98.4	94.5	93.6
Lime	95.2	86.3	96.2	95.3	96.4	94.3	82.3	97.5	94.4	96.5
Parzen	90.5	83.8	91.2	90.1	85.9	88.3	75.3	88.2	87.3	84.6

	Kitchen				
	LR	NN	RF	SVM	Tree
Shap	96.6	89.2	99.4	96.5	94.2
Lime	96.5	88.0	98.3	96.2	96.4
Parzen	91.5	82.6	90.6	89.7	86.6

Table 5: Experiment 5.3 | Real-world data evaluated on precision

Recall (in %)

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	98.5	97.2	98.4	97.9	95.6	98.5	93.1	98.7	98.0	96.2
Lime	97.9	97.5	98.1	97.5	98.3	98.2	92.4	98.7	97.8	98.9
Parzen	79.4	93.0	99.2	90.2	99.6	75.6	81.3	76.7	76.1	100.0

	Kitchen				
	LR	NN	RF	SVM	Tree
Shap	99.0	93.7	99.0	98.7	96.0
Lime	98.8	93.8	99.2	98.8	98.9
Parzen	62.8	89.1	98.2	100.0	81.5

Table 6: Experiment 5.3 | Real-world data evaluated on recall

Accuracy (in %)

	Books					DVDs				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	99.8*	100.0
Lime	99.2	91.8	96.8	98.5	100.0	99.2	92.2	99.2	99.0	100.0
Parzen	63.5	69.2	62.7	72.8	59.0	65.8	60.8	58.2	68.2	53.2

	Kitchen				
	LR	NN	RF	SVM	Tree
Shap	100.0	100.0	100.0	100.0	100.0
Lime	99.8	91.2	99.5	100.0	100.0
Parzen	46.8	79.8	68.5	50.7	69.8

* Shap's accuracy is slightly below 100% despite local accuracy guarantees due to a rounding error on the 16th decimal for a single instance.

Table 7: Experiment 5.3 | Real-world data evaluated on accuracy

Experiment 5.3: Synthetic data

The individual scores for precision, recall and accuracy for Experiment 5.3 on the synthetic data are presented in the tables below.

Precision (in %)

	Redundancy: 0 Noise: 0.05					Redundancy:15 Noise: 0.05				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	96.6	91.2	98.2	95.6	88.5	97.0	93.7	97.4	96.8	87.9
Lime	97.2	91.3	95.2	97.5	89.3	97.1	93.8	96.7	96.3	88.2
Parzen	89.8	89.0	92.1	88.0	87.4	93.5	92.4	95.0	95.0	87.2
	Redundancy: 0 Noise: 0.30					Redundancy:15 Noise: 0.30				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	96.6	81.5	98.3	96.7	83.3	97.4	88.3	97.5	94.8	82.6
Lime	95.4	80.6	95.1	95.8	81.9	94.7	87.6	96.3	93.5	81.5
Parzen	87.4	77.9	90.9	88.4	81.4	90.7	86.6	93.8	91.8	80.8

Table 8: Experiment 5.3 | Synthetic data evaluated on precision

Recall (in %)

	Redundancy: 0 Noise: 0.05					Redundancy:15 Noise: 0.05				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	99.9	98.8	99.0	99.3	100.0	99.7	99.1	99.3	99.6	99.8
Lime	97.1	93.7	95.2	95.1	93.7	97.4	95.5	96.3	95.2	94.9
Parzen	95.9	97.6	96.5	97.0	94.5	98.6	98.3	98.7	97.8	97.3
	Redundancy: 0 Noise: 0.30					Redundancy:15 Noise: 0.30				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	99.3	96.5	97.6	98.5	99.4	98.7	98.0	99.2	98.6	99.7
Lime	95.3	92.4	93.1	95.1	89.4	95.6	94.7	96.7	94.9	91.9
Parzen	93.5	96.0	93.8	96.4	92.7	97.8	97.9	98.9	97.0	96.3

Table 9: Experiment 5.3 | Synthetic data evaluated on recall

Accuracy (in %)

	Redundancy: 0 Noise: 0.05					Redundancy:15 Noise: 0.05				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Lime	97.2	91.8	92.2	96.0	79.8	98.2	90.8	94.8	94.5	84.2
Parzen	88.0	85.8	85.5	81.8	78.8	88.8	91.2	92.2	92.0	79.0
	Redundancy: 0 Noise: 0.30					Redundancy:15 Noise: 0.30				
	LR	NN	RF	SVM	Tree	LR	NN	RF	SVM	Tree
Shap	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0	100.0
Lime	96.0	81.5	92.5	94.8	73.5	93.5	86.2	94.0	89.5	70.2
Parzen	81.2	75.0	82.2	78.0	67.8	81.5	87.0	90.0	91.2	62.3

Table 10: Experiment 5.3 | Synthetic data evaluated on accuracy